**Teste Técnico para Desenvolvedores Back-end**

## Requisitos Gerais:

Você está desenvolvendo um sistema de gerenciamento de tarefas de manutenção realizadas ao longo de um dia de trabalho. A aplicação possui dois tipos de usuários: **Gerente** e **Técnico**.

- **Técnico**: realiza e gerencia suas próprias tarefas. Pode criar, visualizar e atualizar suas tarefas.
- **Gerente**: pode visualizar e deletar tarefas de todos os técnicos e deve ser notificado quando qualquer técnico realiza uma nova tarefa.

Cada tarefa possui:

- Um resumo (máximo de 2500 caracteres).
- A data de realização da tarefa.
- O resumo da tarefa pode conter informações pessoais.

## Notas:

- Esse teste se aplica a todos os níveis de desenvolvedores, então demonstre seu nível.

---

## Funcionalidades a Desenvolver:

1. **API para criação de tarefas**: Endpoint para criar uma nova tarefa.
2. **API para listagem de tarefas**: Endpoint para listar todas as tarefas.
    - Técnicos só podem visualizar suas próprias tarefas.
    - Gerentes podem visualizar tarefas de todos os técnicos.
3. **Notificação ao gerente**: Toda vez que um técnico criar uma nova tarefa, o gerente deve ser notificado (exemplo: “O técnico X realizou a tarefa Y na data Z”).
    - **Desenvolvedores mais experientes**: Implementar essa notificação de forma assíncrona, utilizando um mecanismo que não bloqueie a resposta HTTP (ex: filas, workers, etc.).
4. **Autenticação e Autorização**:
    - Adicionar uma camada de autenticação JWT ou OAuth2 para garantir que apenas usuários autenticados possam acessar a API.
    - Garantir que técnicos só possam visualizar suas próprias tarefas e que os gerentes tenham acesso a todas.

---

## Requisitos Técnicos:

- **Linguagem**: Você pode escolher qualquer linguagem de programação back-end para implementar a solução.
- **Banco de Dados**: Utilize um banco de dados relacional à sua escolha para armazenar os dados da aplicação.
- **Testes**: Adicione testes unitários para garantir o funcionamento correto das funcionalidades.
- **Adicional (opcional)**: Crie um ambiente de desenvolvimento local utilizando **Docker**.

---

## Bônus (Desenvolvedores mais experientes):

- Utilize um **message broker** (ex: RabbitMQ, Kafka) para desacoplar a lógica de notificação do fluxo principal da aplicação.
- Crie arquivos de objetos Kubernetes para deployar a aplicação (Deployments, Services, Secrets, ConfigMaps, etc.).
- Implemente **testes de integração** para validar o fluxo completo da aplicação.
- Adicione suporte para logs estruturados e rastreamento distribuído (ex: OpenTelemetry).

---

## Diretrizes Adicionais (Para Todos os Níveis):

**Configuração e Setup:**

- **Setup Completo e Funcional**: Certifique-se de que o projeto esteja funcional e fácil de configurar desde o início.
- **Variáveis de Ambiente**: Utilize variáveis de ambiente para configurar o banco de dados e outras configurações importantes. Evite hardcoding de valores.
- **Migração de Banco de Dados**: Inclua scripts ou ferramentas para realizar as migrações de banco de dados automaticamente.

**Funcionalidades Principais da API:**

- **Autenticação**: Implemente mecanismos robustos de autenticação (ex: JWT ou sessões).
- **Validação**: Valide todas as entradas adequadamente (ex: tamanho do resumo, tipos de dados).
- **Formatos de Resposta**: Certifique-se de que as respostas da API estejam no formato correto (ex: JSON com o `content-type` apropriado).
- **Tratamento de Erros**: Implemente um tratamento de erros robusto em todos os endpoints.

**Boas Práticas de Segurança:**

- **Gerenciamento de Senhas**: Utilize hashing seguro para senhas (ex: bcrypt).
- **Dados Sensíveis**: Criptografe dados sensíveis quando necessário.
- **Variáveis de Ambiente**: Armazene configurações sensíveis em variáveis de ambiente, e não no código.

**Processamento Assíncrono e Notificações:**

- **Notificações Assíncronas**: As notificações devem ser tratadas de forma assíncrona, utilizando um message broker (ex: RabbitMQ, Kafka).
- **Operações Atômicas**: Certifique-se de que as operações sejam atômicas, especialmente ao atualizar tarefas e enviar notificações. Considere o uso de transações quando necessário.

**Qualidade e Estrutura de Código:**

- **Camada de Controladores**: Use uma camada de controladores dedicada para tratar as requisições HTTP.
- **Camada de Serviços**: Separe a lógica de negócios em uma camada de serviços para manter o código limpo e modular.
- **Código Limpo**: Escreva um código limpo, legível e bem documentado. Siga as melhores práticas de organização e estilo.

**Conteinerização e Deploy:**

- **Dockerização**: Containerize tanto a aplicação quanto o banco de dados utilizando Docker. Garanta que o setup com Docker esteja completo e funcional.
- **Docker Compose**: Utilize Docker Compose para gerenciar configurações de múltiplos contêineres.
- **Kubernetes**: Forneça arquivos de objetos Kubernetes para o deploy, incluindo Deployment, Service, ConfigMap e Secret.

**Testes:**

- **Testes Unitários**: Escreva testes unitários abrangentes para garantir que todas as funcionalidades funcionem conforme o esperado.
- **Testes de Integração**: Inclua testes de integração para verificar se os diferentes componentes da aplicação funcionam corretamente em conjunto.

---

## Armadilhas Comuns a Evitar:

- **Configuração Docker Incompleta**: Garanta que o docker-compose aguarde o banco de dados estar pronto antes de iniciar a aplicação.
- **Configurações Hardcoded**: Evite valores hardcoded. Use variáveis de ambiente para configurações.

- **Falta de Migrações de Banco de Dados**: Inclua scripts de migração para configurar o esquema do banco de dados automaticamente.
- **Documentação Incompleta**: Documente totalmente os endpoints da API utilizando ferramentas como Swagger/OpenAPI.
- **Falta de Autenticação**: Certifique-se de que mecanismos robustos de autenticação estejam implementados.
- **Validação Inadequada de Entradas**: Valide todas as entradas para evitar vulnerabilidades de segurança e garantir a integridade dos dados.
- **Notificações Síncronas**: Evite notificações síncronas que bloqueiem as requisições HTTP.
- **Credenciais em Texto Plano**: Nunca guarde credenciais em texto plano. Utilize métodos seguros para gerenciar segredos.